On the Effects of Programming and Testing Skills on External Quality and Productivity in a Test-Driven Development...

Conference Paper · April 2014

DOI: 10.1145/2745802.2745826

CITATIONS
5

READS
158

3 authors:

Davide Fucci
University of Hamburg
19 PUBLICATIONS **43** CITATIONS
SEE PROFILE

Burak Turhan
Brunel University London
148 PUBLICATIONS **1,358** CITATIONS
SEE PROFILE

Markku Oivo
University of Oulu
139 PUBLICATIONS **698** CITATIONS
SEE PROFILE

On the Effects of Programming and Testing Skills on External Quality and Productivity in a Test-Driven Development Context

Davide Fucci
University of Oulu
M-Group
Oulu, Finland
davide.fucci@oulu.fi

Burak Turhan
University of Oulu
M-Group
Oulu, Finland
burak.turhan@oulu.fi

Markku Oivo
University of Oulu
M-Group
Oulu, Finland
markku.oivo@oulu.fi

ABSTRACT

Background: In previous studies, a model was proposed that investigated how the developers' unit testing effort impacted their productivity as well as the external quality of the software they developed.

Goal: The aim of this study is to enhance the proposed model by considering two additional factors related to the expertise of developers: programming and unit testing skills. The possibility of including such skills in a model that represents the relationship that testing effort has with the developer's productivity and the product's external quality was investigated.

Method: Data collected from a test-first development task in academic setting was used in order to gauge the relationship between testing effort, external quality, and productivity. Furthermore, Analysis of Covariance (ANCOVA) was utilized to check the impact of developers' skills on productivity and quality.

Result: The results obtained in previous studies were confirmed: there exists a positive effect of testing effort on productivity, but not on quality. Moreover, the developers' skills have an impact on productivity but none on external quality.

Conclusion: Productivity improves with testing effort, a result consistent across previous, similar studies. The role of existing skills is a relevant factor in studying the effects of developers' unit testing effort on productivity. Nevertheless, more investigations are needed regarding the relationship between unit testing effort and external quality.

Keywords

unit testing, external quality, developers' productivity, developers' skills

1. INTRODUCTION

Unit testing is defined as *"a test, executed by the developer in a laboratory environment, that should demonstrate that the program meets the requirements set in the design specification"* [12]. Hence, unit tests are created by the same developer who writes the production code. In object orientation, the *program* (or unit) under test can vary in granularity, from something as broad as an interface or class to a single method although what is recommended is to treat a specific behaviour which may be comprised of several methods, as a unit. [11]. With the advent of extreme programming and test-driven development (TDD) [3], unit testing has become more important in the arsenal of a software developer as well as in computer science and software engineering academic curricula [2]. In our previous work [8], we investigated the relationship between the developers' unit testing effort and the external quality (e.g., number of defects) of the software system they produced, as well as the relationship between the developers' testing effort and their productivity. The baseline model was initially proposed by Erdogmus et al. [6] in the context of a study on the effects of test-first development (TF). In particular, Erdogmus et al. claimed that developers using TF put more effort into unit testing, i.e., producing more unit tests compared to an iterative development methodology in which tests are written after production code, referred to as test-last development (TL) [6]. In turn, they hypothesized that such testing effort has a positive impact on the quality and productivity of the developers. However, Erdogmus et al. and our previous study showed a significant relationship between testing effort and productivity, the results regarding external quality were inconclusive in both cases. However, neither study took into account the skill-set necessary to properly apply TF or TL. As such, we hypothesise that two basic skills are needed to be taken into account for the application of both development methodologies: the programming language and the unit testing skill. In this study, our aim is to improve the model proposed by Erdgomus et al. by including such skills. The research question investigated in this study was the following: *Do the developers' skills have an effect on the relationship between testing effort and the external quality of the system and developers' productivity?* To answer this research question, we investigated an enhanced version of the model proposed in Erdogmus et al. which we tested using data collected from the participants of a graduate university course on software testing.

2. RELATED WORK

Table 1: Original study and replication contexts

Context variable	Erdogmus et al., 2005	Fucci and Turhan, 2013
Subject type	46 undergraduate (35 after drop-outs)	50 mixed graduate and undergraduate (30 after drop-outs)
Subject unit	Individuals	Individuals and Pairs
Development environment	Java, Eclipse, JUnit	Java, Eclipse, JUnit
Experiment task	Robert Martin's Bowling Scorekeeper	Robert Martin's Bowling Scorekeeper
Task type	Fine grained, incremental difficulty	Fine grained, incremental difficulty
Time to complete the task	Several lab sessions, remote work	Single lab session (3 hours)
Outcomes under study	$TESTS, PROD, QLTY$	$TEST, PROD, QLTY$

Many of the studies that have investigated test-driven development [3] have implicitly addressed a particular scenario in which unit tests are performed before writing the production code. Hence, we reviewed the following TDD literature with the awareness that TDD is one particular context in which unit testing is heavily applied. For example, the study of Müller and Höfer [17] compared expert and novice developers using TDD in a controlled experiment. The authors' goal was to examine the effects of knowledge and skill on several outcomes (e.g., code coverage, code quality). The results showed that there are differences between the two groups. In particular, the experts write better code (i.e., superior internal quality and coverage), although there are no differences in terms of test-to-production code ratio. The study suggested that the findings of TDD experiments involving students are difficult to generalise since there is a remarkable gap between the way professionals and novices use TDD. A quasi-experiment by Höfer and Philipp [10] found that the most experienced subjects applying TDD performed better, in terms of code refactoring and internal quality, than novices applying the same technique. On the other hand, more expert developers (working in pairs) achieved a lower productivity than the others. When synthesising the results of experiments carried out with subjects that had different skill levels (e.g., academic vs. industrial experiments), Rafique et al. [19] showed that quality was improved by the employment of TDD in industrial settings, although with a drop in productivity when compared to academia. Interestingly, the lack of unit testing experience has been identified as a limiting factor in the adoption of TDD by the industry in a systematic literature review by Causevic et al. [5]. In particular, the authors found that—among studies performed with students—the subjects found it difficult to write good test cases, hence limiting the validity of such studies. They also suggested that adopting TDD without proper training or testing skill can be risky [5]. A longitudinal study by Latorre [13] investigated the effects of developers' knowledge and experience on the application of TDD over one month. The author concluded that skilled developers, with appropriate knowledge of unit testing, are able to properly apply TDD after a short training period. Furthermore, such ability was retained over time. At the same time, less skilled developers had a steeper learning curve; nonetheless, their ability with TDD reached the same level as the experts by the end of the study. Despite such results, differences remain between experts and novices in terms of design. This, in turn, reduces novices' productivity because the time required to fix their design [13]. Factors such as existing skills and experience are also taken into account in a study by Madeyski [14] to control for the effects of applying unit testing, in both traditional and test-first fashion, during an experiment with students. The author investigated the impact of the different testing techniques on quality measures like branch coverage and mutation score. Neither the testing technique or pre-existing skills and experience of the subjects showed a significant effect. A survey of existing literature demonstrated interesting results considering the few studies that specifically address how unit testing effort impacts the quality of a system [15]. The authors conducted a multiple industrial case study in which they showed that the effort spent in creating unit-test cases is a good predictor for code coverage. While coverage cannot be directly associated with the quality of the system, it is interesting to note that the authors observed that the level of achievable coverage is not linearly correlated with unit testing effort. More specifically, while reaching 50% of coverage does not require much effort, levels over 90% might not be feasible. Finally, the study from which we derived our model (Erdogmus et al. [6]) was composed of two stages. The *first stage* is a controlled experiment [21] comparing test-driven development to iterative test-last development, using students as subjects. The *second stage* was a non-experimental study in which the unit testing effort was used to model two dependent variables (external quality and productivity) independently from the experimental groups in the first stage. The rationale was that the testing effort might be a proxy measure of the claimed effects of TDD (improved quality and productivity), due to its central role in this technique [6]. In other words, the authors claim that, since TDD developers are believed to put more effort in unit testing when compared to test-last developers, the effects on external quality and productivity are likely to become evident when an high level of testing effort is observed. We previously conducted a study in which we focused on replicating the second stage of Erdogmus et al. [8]. The results were consistent between the two studies, in spite of the changes that were made to the context in our replication. The context of the two studies are reported in Table 1. The original study hypotheses are summarized in Table 2. Please note that we did not test those hypotheses in this study, as they were tested in our previous works [7, 9]. In this work, we add a new set of hypotheses to enhance the models tested in the second stage of the Erdogmus et al. study.

3. DEVELOPERS' SKILLS FACTOR

In this study, we introduced two factors dealing with developers' skills, which were not taken into account in the model formulated in Erdogmus et al. or in its replication [8]. One limitation of Erdogmus et al. is that the subjects, although all were characterized as *undergraduate*, might have had different programming and testing skills that were not taken into account during the execution of the study [6]. This is how we differentiated and extended the original study. In this study, we gathered information about the subjects'

Table 2: Formalized hypotheses for the models in Erdogmus et al.
(β_1 represents the regression coefficient)

Name	H_0	H_1
Q	$QLTY = \beta_0 + \beta_1 \times TEST, \beta_1 = 0$	$QLTY = \beta_0 + \beta_1 \times TEST, \beta_1 \neq 0$
P	$PROD = \beta_0 + \beta_1 \times TEST, \beta_1 = 0$	$PROD = \beta_0 + \beta_1 \times TEST, \beta_1 \neq 0$

pre-existing skills in the Java programming language (i.e., the programming language used by the participants) and unit testing through a questionnaire. Erdogmus et al. devised two regression models, one for external quality and one for productivity, to quantify the relationship between testing effort and either of the two outcomes. The models are formally expressed using statistical hypothesis testing in Table 2, where hypothesis Q deals with external quality and hpyothesis P deals with productivity. We formulated four hypotheses to re-assess the regression models in order to take into account the developers' skills. The hypotheses were formulated as follows (note that the models are reported in Table 2 :

Q(JAVA) - Does the Java skill interact with the model Q?
P(JAVA) - Does the Java skill interact with the model P?
Q(UT) - Does the unit testing skill interact with the model Q?
P(UT) - Does the unit testing skill interact with the model P?

The metrics used for the hypotheses formulation are described in the following section.

3.1 Metrics

The metrics used to gauge external quality ($QLTY$), programmers' productivity ($PROD$) and testing effort ($TEST$) were calculated following the formulas used by Erdogmus et al. [6]. The newly introduced metrics, $JAVA$ and UT, were gauged using a four-point Likert item for each variable. In particular, the subjects were asked to select one answer among *None*, *Fair*, *Good* and *Excellent* for the statements:

JAVA – "Rate your skill with the Java programming language".
UT – "Rate your skill with unit testing".

Since all the participants in the course were required to answer the questionnaire, our response rate was 100%.

3.2 Context, Subjects and Procedure

This study took place in academic setting. The subjects we sampled were participants in the *Software Quality and Testing* lab course, offered as part of the master's programme in Information Processing Science at the University of Oulu, Finland during Fall 2013. Hence, the sampling was done by convenience. The course consisted of theoretical classes, as well as seven hands-on laboratory sessions that were each three hours long. During the first six sessions, the subjects were introduced to unit testing and test-driven development in Java. The subjects worked in pairs or solo through the sessions, solving several code katas while interacting with the instructor to discuss testing, object-oriented design and refactorings. We administered a pre-questionnaire to the subjects during the first lab session. The pre-questionnaire contained the items we used to measure the subjects' self-assessed skills regarding the Java programming language, unit testing and the tools and frameworks used during the course. We collected the code artefacts the subject had produced during the last session, when they were asked to complete a programming exercise, working solo and without any interaction with their peers or the instructors. The exercise used was Robert Martin's Bowling Scorekeeper, composed of 13 fine-grained user stories. In order to have the same APIs for all the subjects, we provided a stub project containing only the method signatures (30 SLOC). We recorded the time the subjects used to complete the task and administered a post-questionnaire to gauge their feedback on the usefulness and difficulty of the concepts they had been introduced to during the course. Once the session was over, the remaining subjects were asked to stop working, return their code artefacts and fill the post-questionnaire. Forty-one students participated in the experimental session; therefore, we collected and inspected their 41 code artefacts and discarded 8 that were either empty or that failed to run the tests or to compile. Our dataset, composed of 33 data points, was constructed by extracting the relevant metrics from the artefacts (SLOC mean = 120, min = 58, max = 301, sd = 34).

4. RESULTS

In this section, we present the results of the statistical analysis. We first describe the dataset and then check the impact of subjects' skills on the models proposed in Table 2.

4.1 Descriptive Statistics

The descriptive statistics for the continuous variables in our dataset ($QLTY$, $PROD$ and $TEST$) are reported in Table 3. Although the majority of the subjects tend to accumulate towards the centre mean value (mean = 90.92, sd = 6.26), the distribution of the $QLTY$ variable appears to be bimodal. This is in line with our previous experiences whereby we already observed such *sparsity* [7, 8]. The distribution of $PROD$, on the other hand, appears to be unimodal and right-skewed. The distribution of $TEST$ also appears to be unimodal and right-skewed, mostly due to the presence of extreme values. Nevertheless, such extreme values are genuine and not due to measurement errors and should be taken into further account during the analysis. Regarding the measure of the subjects' skills, the results of the pre-questionnaire—filled in before the experiment—show that the average programming experience is 1.5 years. Five subjects declared themselves to be junior professional software developers, three of them declared that they only had one year of professional experience, one declared having two years of experience and the most experienced one have been working as a professional software developer for four years. The resulting measurement for $JAVA$ and UT—using two 4-point Likert scale items in the pre-questionnaire—is reported in Table 4. According to Table 4, none of the subjects declared that they had *Excellent* unit testing skills, while almost everyone claims to have no greater than *Fair* unit testing skills. On the other hand, the majority of the responses regarding Java programming skills were between *Fair* and *Good*. The majority of the subjects falls in the intersection between *Fair* Java programming skills and *None* unit

Table 3: Descriptive statistics for the continuous variables under study ($n = 33$)

	mean	sd	median	min	max	range	1st Qu.	3rd Qu.
QLTY	88.41	6.31	88.54	76.67	100.00	23.33	82.67	90.37
PROD	5.58	3.97	4.44	1.11	13.85	12.74	2.78	8.80
TEST	8.30	6.22	7.22	0.56	20.00	19.44	2.50	12.50

Table 4: Pre-questionnaire results for JAVA and UT

	JAVA	UT
None (0)	2	19
Fair (1)	18	10
Good (2)	11	4
Excellent (3)	2	0

testing skills. Congruently, the only two subjects declaring *None* for Java skills declared the same for unit testing. Out of the two subjects that declated *Excellent* for Java skills, one declared to possess *Fair* unit testing skills, whereas the other declared his or her skills as *None*.

4.1.1 Assumption testing

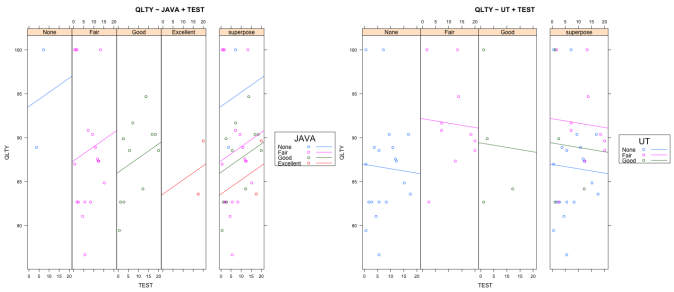
We use analysis of covariance (ANCOVA) in order to create the models necessary to evaluate our hypotheses. Confidence in the results depended on the degree to which our data satisfied the assumptions underlying the models. Hence, we first checked wether the collected data followed the ANCOVA assumptions. Thereafter, we determined that our data met the assumptions that observations should be independent and that they must be measured on an interval or ratio scale. The normality assumptions for the outcomes (dependent variables) is not met according to the results of the Shapiro-Wilk test for both $QLTY$ ($p-value = .03$) and $PROD$ ($p-value = .8e-03$). Further, we checked for multicollinearity among the two ordinal variables $JAVA$ and UT. The Spearman's rank correlation coefficient and associated $p - value$ showed a non-significant correlation ($\rho = .313$, $p - value = .080$). Singularity, i.e., one variable being a combination of other variables, was not an issue in our measurements (See section 3.1). Moreover, the ANCOVA design assumed homogeneity of variance within the four groups for $JAVA$ and UT. The result of Bartlett's test, not reported for sake of brevity, showed that the assumptions held. The ANCOVA designs assumed homogeneity of regression slopes. In this case, it was assumed that the regression slope representing the relationship between $QLTY$ and $TEST$ was the same in each of the four skills' levels (for both $JAVA$ and UT). A test for the homogeneity of regression slopes can be obtained by including an interaction term in the model. We had two possible interaction terms for the two outcomes: an interaction between $TEST$ and $JAVA$ ($TEST : JAVA$) and between $TEST$ and UT ($TEST : UT$). A significant interaction would have implied that the relationship between testing effort and external quality or productivity depend on the level of the skills. However, the F-test results for the interaction of regression slopes, not reported for the sake of brevity, showed that none of the interactions terms were significant, supporting the assumption of equality of slopes.

4.2 Regression Model Analysis

We analyzed the regression models in Q and P (see Table 2), controlling for the two skill metrics, $JAVA$ and UT, by means of ANCOVA. Thus, we built four models, which are presented in Fig. 1 and Fig. 2. The first model (Fig. 1a) revealed no effects of $TEST$ ($F(1, 28) = 0.60$, $p - value = 0.44$) or $JAVA$ ($F(3, 28) = 0.70$, $p - value = 0.55$) on $QLTY$. The model (Fig. 1b) predicting an effect of $TEST$ on $QLTY$ was not significant ($F(1, 29) = 0.68$, $p - value = 0.79$), nor was the effect of UT ($F(2, 29) = 2.1$, $p - value = 0.13$). In the model presented in Fig. 1a, the *None* and *Excellent* groups were ignored, since they include only two data points. The slopes are constant among the two remaining groups (i.e., there was no significant interaction between $JAVA$ and $TEST$), and the intercepts showed counterintuitive results. The *Fair* group achieved a slightly higher level of productivity than the *Good* group. When UT is taken into account, as in Fig. 1b, the *Fair* group achieved better quality than the *None* group. The *Good* group was left out of Fig. 1b due to the the low number of data points, whereas the *Excellent* group was not shown since none of the subjects claimed to possess such a level of unit testing skill. Thus, we conclude that there is no relationship between testing effort and external quality, when controlling for either Java or unit testing skills of the subjects. *The null hypothesis of Q(JAVA) and Q(UT) fails to be rejected.* The ANCOVA F-test for $PROD$ indicated that $JAVA$ is correlated to $PROD$ ($F(3, 28) = 8.20$, $p - value << 0.01$) and $TEST$ is also correlated to $PROD$ after controlling for $JAVA$ ($F(1, 28) = 23.21$, $p - value << 0.01$). Moreover, the correlation between $TEST$ and $PROD$ was not the same for the $JAVA$ groups. Specially, the mean values obtained after controlling for the effects of $JAVA$ were 3.55 for the *None* group, 4.39 for the *Fair* group, 6.47 for the *Good* group and 8.48 for the *Excellent* group. Fig. 2a shows the increasing level of $PROD$ among the different levels. If the *None* and *Excellent* groups were left out, since their models were based on only two data points each, the baseline for the correlation increases with the increment of Java programming skill. Hence, the Java programming skills of the subjects had an effect on the model showing that the testing effort had a relationship with the productivity of the developers. Fig.2b represents the productivity model includes unit testing skills of the subjects. The ANCOVA F-test indicated that there was a significant relationship between $TEST$ and $PROD$ ($F(1, 29) = 32.98$, $p - value << 0.01$), controlling for UT ($F(1, 29) = 3.45$, $p - value = 0.04$). The adjusted means demonstrate that the subjects with *Good* skills achieved better productivity ($mean = 6.64$) than subjects in the *Fair* group ($mean = 5.84$) and the *None* group ($mean = 5.22$). *The null hypothesis in P(JAVA) and P(UT) are thus rejected.*

5. THREATS TO VALIDITY

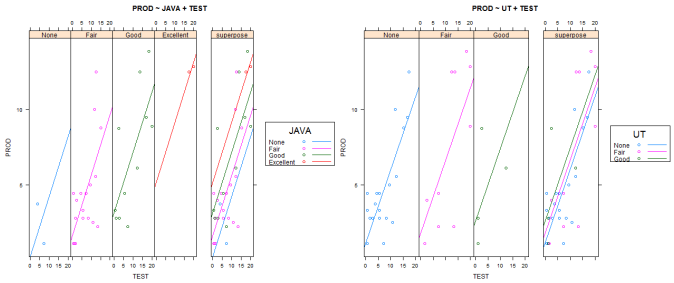
The main threat to the validity of our study is the biased population (*university students*) from which the sample was drawn when the reference population is *software developers*. Nevertheless, other studies have showed that the more skilled student can perform at the same level of professional



(a) TEST vs. QLTY faceted by JAVA

(b) TEST vs. QLTY faceted by UT

Figure 1: Representations of the external quality model using skill as a covariate. Confidence interval 95%.



(a) TEST vs. PROD faceted by JAVA

(b) TEST vs. PROD faceted by UT

Figure 2: Representations of the developers' productivity model using skill as a covariate. Confidence interval 95%.

Table 5: Summary of ANOVA for *TEST*

JAVA	Estimate	Std. Error	$p - value$
None	-13.26	8.56	.13
Fair	-11.83	6.38	.07
Good	-6.50	6.54	.32
UT	Estimate	Std. Error	p-value
None	-4.26	3.46	.22
Good	.93	4.85	.84

developers [17, 18], hence limiting the validity threat to our study. The 8 of the 41 artefacts discarded is a threat to the interval validity, and this can be attributed to the difficulty of the task. In fact, from an analysis of the post-questionnaire, we can see that the task was perceived as being difficult to apply by 70% of the respondents, although some subjects stated that its fine granularity (as in the case of the experimental task) eased the testing process. It is necessary to mention that the research hypotheses and the expectations of the researchers were not disclosed to the subjects. Furthermore, process conformance is a threat to validity that was controlled by periodically reminding the subjects to follow the testing practices. Redarding internal validity, our work suffered from mono-operation and mono-method biased, since we studied the constructs using only a single task and measured each of them with only a single metric. At the same time, there might have been other constructs (i.e., internal code quality) that might have been affected, other than quality and productivity, though they were not part of the observed variables. The Hawthorne effect [1] might have taken place, since the subjects were observed all the time during the experiment. Additionally, the conclusion validity of the study is threatened by the low statistical power we were able to achieve in the data analysis due to the limited sample size [16]. The subjects should be homogeneous by construct, since they were sampled from university students who are in the same curricula. Finally, the task complexity and its duration is not a good representation of the real world. However, given the statistical limitations of our study, we do not claim to have achieved generalisable results; rather we regard this kind of study as *explorative*, offering us an affordable opportunity to investigate constructs such as skills, and further validate (or invalidate) their effects through replications in a real world context.

6. CONCLUSION AND FUTURE WORK

In this study, we analyzed the impact of subjects' existing skills on the regression models presented in Erdogmus et al. [6] and Fucci and Turhan [8]. In particular, the models postulated a linear relation between testing effort and the two outcomes, external code quality and productivity. We found that the results provided by these previous studies still hold throughout different levels of Java programming and unit testing skills. In particular, the model for productivity is significant, but not the one for quality. Our results also show that, although the correlation is stronger for the subjects with higher pre-existing skills, focusing on testing (either in a TDD or test-last fashion) can impact the productivity of the less skilled subjects as well. This might contradict the claim that unit testing, particularly its application to test-driven development, has a steep learning

curve [20]. The data we collected shows that Java programming skill has an interaction with the linear model that relates testing effort and productivity as stated in Hypothesis $P(JAVA)$. Similarly, unit testing skill has an interaction with the model that relates testing effort to productivity (Hypothesis $P(UT)$). However, we could not find a significant relationship between testing effort and external quality. This result holds throughout the three studies [6, 8, 7]. We checked whether the pre-existing skills had an impact on the quality model, but we did not find any for either Java programming or unit testing skills. This raises the question of whether testing effort is a good explanatory variable or if there are other variables, that have stronger effect, that are have not been taken into account. Hence, we conclude that Java skill does not interact with the model that relates testing effort and external quality. Similarly, unit testing skills do not interact with the model that relates testing effort with external quality. Although we advise that further studies are necessary, for example, to gauge the impact of training the subjects on the models presented in this study, the answer to the research question is the following: *Developers' skills, like programming language and unit testing, have a significant impact on developers' productivity but not on the external quality of the developed software.* Future studies should aim to access subjects whose skills can vary sensibly and aim to improve external validity. A major improvement to this study would be to measure the constructs with different metrics. For example, when gauging testing effort, the quality of the test should be taken into account [4]. Finally, the material necessary for further replications is available from the first author.

ACKNOWLEDGMENT

This research is partially supported by the Academy of Finland with decision no: 278354. The first author would like to acknowledge ISACA Finland Chapter for the support provided to complete this work.

7. REFERENCES

[1] J. G. Adair. The Hawthorne effect: A reconsideration of the methodological artifact. *Journal of Applied Psychology*, 69(2):334, 1984.

[2] T. Astigarraga, E. Dow, C. Lara, R. Prewitt, and M. Ward. The emerging role of software testing in curricula. In *Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, 2010 IEEE*, pages 1–26, April 2010.

[3] K. Beck. *Test-driven Development: by Example*. The Addison-Wesley signature series. Addison-Wesley, 2003.

[4] A. Causevic. Test case quality in test driven development: a study design and a pilot experiment. *IET Conference Proceedings*, pages 223–227(4), January 2012.

[5] A. Causevic, D. Sundmark, and S. Punnekkat. Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review. In *Fourth International Conference on Software Testing, Verification and Validation (ICST)*, pages 337–346. IEEE.

[6] H. Erdogmus, M. Morisio, and M. Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, 2005.

[7] D. Fucci and B. Turhan. A Replicated Experiment on the Effectiveness of Test-First Development. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 103–112. IEEE, 2013.

[8] D. Fucci and B. Turhan. On the role of tests in test-driven development: a differentiated and partial replication. *Empirical Software Engineering*, pages 1–26, 2013.

[9] D. Fucci, B. Turhan, and M. Oivo. Conformance Factor in Test-driven Development: Initial Results from an Enhanced Replication. In *2014 ACM/IEEE International Conference on Evaluation and Assessment of Software Engineering (EASE)*. IEEE, 2014.

[10] A. Höfer and M. Philipp. An empirical study on the tdd conformance of novice and expert pair programmers. In *Agile Processes in Software Engineering and Extreme Programming*, pages 33–42. Springer, 2009.

[11] D. Huizinga and A. Kolawa. *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Pr, 2007.

[12] T. Koomen and M. Pol. *Test Process Improvement: A practical step-by-step guide to structured testing*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[13] R. Latorre. Effects of developer experience on learning and applying Unit Test-Driven Development. *Transactions on Software Engineering*, 40(4):381–395face, 2014.

[14] L. Madeyski. The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment. *Inf. Softw. Technol.*, 52(2):169–184, 2010.

[15] A. Mockus, N. Nagappan, and T. Dinh-Trong. Test coverage and post-verification defects: A multiple case study. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 291–301, Oct 2009.

[16] A. Montenegro. On sample size and precision in ordinary least squares. *Journal of Applied Statistics*, 28(5):603–605, 2001.

[17] M. M. Müller and A. Höfer. The Effect of Experience on the Test-driven Development Process. *Empirical Software Engineering*, 12(6):593–615, 2007.

[18] M. Philipp. Comparison Of The Test-Driven Development Processes Of Novice And Expert Programmer Pairs. 2009.

[19] Y. Rafique and V. Misic. The effects of test-driven development on external quality and productivity: a meta-analysis. 2013.

[20] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus. What Do We Know About Test-driven Development? *Software, IEEE*, 27(6):16–19, 2010.

[21] C. Wohlin. *Experimentation in Software Engineering: an Introduction*, volume 6. Springer, 2000.